Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS

Lizhe Wang[†], Gregor von Laszewski[†], Jai Dayal[‡] and Fugang Wang[†]
† Pervasive Technology Institute, Indiana University
‡ Department of Computer Science, Rochester Institute of Technology

Abstract—**Reducing energy consumption for high end computing can bring various benefits such as, reduce operating costs, increase system reliability, and environment respect. This paper aims to develop scheduling heuristics and to present application experience for reducing power consumption of parallel tasks in a cluster with the Dynamic Voltage Frequency Scaling (DVFS) technique. In this paper, formal models are presented for precedence-constrained parallel tasks, DVFS enabled clusters, and energy consumption. This paper studies the slack time for non-critical jobs, extends their execution time and reduces the energy consumption without increasing the task's execution time as a whole. Additionally, Green Service Level Agreement is also considered in this paper. By increasing task execution time within an affordable limit, this paper develops scheduling heuristics to reduce energy consumption of a tasks execution and discusses the relationship between energy consumption and task execution time. Models and scheduling heuristics are examined with a simulation study. Test results justify the design and implementation of proposed energy aware scheduling heuristics in the paper.**

Keywords – Cluster Computing; Green Computing; Task Scheduling

I. Introduction

Nowadays, high end computing facilities can consume a very large amount of power albeit they provide high performance computing solutions for scientific and engineering applications [1]. For example, operating a middle-sized data center (i.e., a university data center) demands 80000kW power [2]. It is estimated that computing resources consume around 0.5% of the world's total power usage [3], and if current demand continues, is projected to quadruple by 2020. Energy consumption for high performance facilities thus contributes to a significant electric bill. Additionally, high power consumption in general results in higher cooling costs. Furthermore, to allow computing facilities to operate on high power for a long time will lead to high temperature of computing systems, which further harms a system's reliability and availability. Therefore, reducing power consumption for high end computing becomes a critical research topic.

Modern processors are equipped with the Dynamic Voltage Frequency Scaling (DVFS) technique, which enables processors to be operated at multiple frequencies under different supply voltages. The DVFS technique thus gives opportunities to reduce the energy consumption of high performance computing by scaling processor supply voltages. Our research is devoted to developing scheduling heuristics which reduce energy consumption of parallel task execution by using the DVFS mechanism. A parallel task is a set of jobs with precedence constraints. Jobs in a parallel task may have some slack time for their execution due to their precedence constraints.

This paper makes a study on scheduling policies and application experiences to reduce power consumption of parallel tasks. An energy aware task scheduling has been developed where we identify the slack time for non-critical jobs and scale their supply voltages thus reducing the jobs' energy consumption.

The green Service Level Agreement (SLA) is introduced in this research. By negotiating with users via green SLA, an energy-performance tradeoff algorithm is developed to reduce energy consumption with an affordable task execution time increase. We develop a simulation study on the proposed scheduling heuristics and make a performance evaluation.We declare our contribution as follows:

- We develop formal models for parallel tasks and a power aware cluster and we also define the task scheduling issue.
- We present the green SLA use scenarios and propose new scheduling heuristics for energy aware parallel task scheduling, which only considers a best effort scheduling scenario, but also makes a study on the tradeoff between the energy consumption and task execution time (performance).
- We build a simulation study and performance evaluation on the proposed heuristics. Test results justify our design and implementation of energy aware heuristics.

The rest of this paper is organized as follows. Section II introduces background and related work. Section III presents the Service Level Agreement with performance metrics of green computing. Then Section IV discusses the models for parallel tasks, DVFS and compute clusters and Section V formally define the research issue of energy aware parallel task scheduling. We propose the scheduling heuristics in Section VI. Section VII describes a simulation study on the proposed scheduling heuristics and finally this paper is summarized in Section VIII.

II. Related Work

This section discusses background and related work of task scheduling, DVFS, and power aware cluster computing.

A. Parallel task scheduling

Task scheduling techniques in parallel and distributed systems have been studied in great detail with the aim of making use of these systems efficiently.

Task scheduling algorithms are typically classified into two subcategories: static scheduling algorithms and dynamic scheduling algorithms. In static task scheduling algorithms, the task assignment to resources is determined before applications are executed. Information about task execution cost and communication time is supposed to be known at compilation time. Static task scheduling algorithms normally are non-preemptive – a task is always running on the resource to which it is assigned [4], [5]. Dynamic task scheduling algorithms normally schedule tasks to resources in the runtime to achieving load balance among PEs. are based on the redistribution [6], [7].

List scheduling algorithm is the most popular algorithm in the static scheduling [8], [9]. List based scheduling algorithms assign priorities to tasks and sort tasks into a list ordered in decreasing priority. Then tasks are scheduled based on the priorities.

In this paper, we build a list based scheduling heuristic for parallel tasks. The task execution information, such as task execution cost and communication cost, can be obtained by some profiling tools and compiler aides in advance.

B. Energy reduction via DVFS technique

Dynamic voltage and frequency scaling (DVFS) has been proven to be a feasible solution to reduce processor power consumption [10], [11]. By lowering processor clock frequency and supply voltage during some time slots, for example, idle or communication phases, large reductions in power consumption can be achieved with only modest performance losses. A DVFS-enabled cluster [1] is a compute cluster where compute nodes can run at multiple power/performance operating points. The DVFS techniques have been applied in the high performance computing fields, for example, in large data centers, to reduce power consumption and achieve high reliability and availability [12], [13], [14]. Popular DVFS-based software solutions for high end computing include:

- Scientific applications can be modeled with Directed Acyclic Graph (DAG) and the critical path is identified in for applications. Thus, it is possible to reduce energy consumption by leveling down the processor supply voltage during non-critical execution of tasks [15].
- Some work [16] builds online performance-driven runtime systems to automatically scale processor supply voltages.
- Some work applies DVFS during the communication phases of high performance computing, for example MPI [17], [18].
- In addition to parallel applications, virtual machine scheduling can also use DVFS [1].

Our research in this paper falls into the first category: scheduling DAGs on multiple processors in a cluster with DVFS techniques.

C. Power aware task scheduling

A lot of work has developed DVFS for task scheduling. For example, [19], [20] discuss scheduling independent tasks with DVFS on a single processor, [21], [22] use DVFS to schedule dependent tasks on multiple processors, [23], [24], [25] developed power aware task scheduling algorithm for real time systems. As our work is devoted to developing power aware scheduling algorithms for dependent tasks, we compare our work with related research in this topic.

[26], [23], [27], [24], [25] schedule dependent tasks on real time, where the tasks normally are assigned with arrival time, deadline and max power consumption. In our research of energy aware high end computing, we don't have these restrictions on the tasks to be scheduled.

[28] employs the similar DAG model and resource model with us and developed energy-aware duplication scheduling algorithms. This work however did not use DVFS technique to reduce power consumption, therefore their implementation certainly has some room to further reduce energy consumption if DVFS technology is employed when scheduling parallel tasks.

[22] proposes a list based low energy scheduling algorithm – LEneS. It smartly introduces enhanced task-graphs (ETG) and energy gain in the list based scheduling. [23] develops a hybrid global/local search optimization framework for DVFS with simulated heating. LPHM [29] is a low power scheduling of DAGs to minimize task execution time. LPHM combines the heterogeneous earliest finish time with the DVFS technique. Compared to the above related research, our work not only considers minimizes the energy consumption in the scheduling algorithm, but also uses the concept of slack time for jobs in power Gantt chart to discuss the trade off between energy consumption and scheduling length.

[30] proposes an energy-conscious scheduling (ECS) heuristic for parallel tasks on heterogeneous computing systems. [31] uses the same idea of extending task execution time by reclaiming slack times for non-critical jobs. In our paper, we give a clear formulation of energy aware scheduling algorithm and a detailed discussion of slack time computation. Our scheduling algorithm also concerns reducing voltages during the communication phases between parallel jobs. None of above research work discusses this aspect.

III. SLA MANAGEMENT FOR GREEN COMPUTING

Green computing is a research topic to make computing with environmental concerns [32], for example, reduced energy consumption and reduced CO_2 emissions. We develop power aware scheduling for parallel task in the context of green SLA (Service Level Agreement for Green Computing). Users can specify not only performance requirements for computing services, but users can also specify green computing requirements for executing their jobs. We define the green SLA in three phases:

- Green SLA contract definition
  Our previous work [32] has summarized a number of green computing metrics, such as Data Center Infrastructure Efficiency (DCiE) [33], [34], Power Usage Effectiveness (PUE) [34], Data Center energy Productivity (DCeP) [35], Space Watts and Performance (SWaP) [36], storage, network, and server utilization. The green SLA contract definition phase creates various green SLA templates based on above green computing metrics. Typical

metrics includes task response time, CO_2 emission, and power consumption. This phase also contains green SLA template publication and discovery.

- green SLA negotiation & monitoring
  Users develop their green SLA specification based on SLA templates and make a negotiation with computing resources, for example, a high performance cluster. Here are some examples of green computing service specifications:
  - Establish an execution service for x minutes if the total carbon emission of the service is below y tons.
  - I would like to accept $z\%$ task execution time increase to reduce w energy consumption.
- green SLA enforcement
  When a green SLA is reached, computing resources then execute the specified green services. For example, schedule tasks based on specified task execution time, CO_2 emission and power consumption. In Section VI, we develop energy aware scheduling algorithms for parallel tasks based on user's green SLA specifications.

Figure 1 shows the conceptual framework for green SLA based on energy aware scheduling in a cluster. Before a resource consumer submits a parallel job to a cluster, she/he firstly negotiates with a resource provider with normal performance metrics, like job response time, as well as with green metrics, for example, power consumption or CO_2 emission. After an agreement is reached, the user then submits his/her job to the resource. The resource provider then schedules the incoming job to an energy aware cluster to guarantee the green metrics and computing performance.

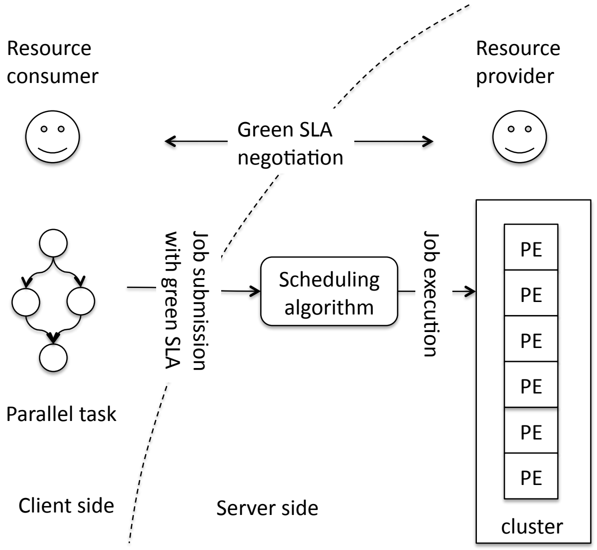


Fig. 1. Concept framework for green SLA based energy aware scheduling in a cluster

IV. System Model

This section provides the formal description for a DVFS-enabled cluster, parallel tasks, and performance models, which are employed as basis of the formal problem definition in Section V and the scheduling algorithm in Section VI.

A. DVFS Model

A DVFS-enabled processor can be operated on a set of supply voltages V and a set of processor frequencies F.

$$V = \bigcup_{1 \le m \le M} \{v_m\} \tag{1}$$

$$F = \bigcup_{1 \le m \le M} \{f_m\} \tag{2}$$

where,
v_m is the m-th processor operating voltage;
f_m is the m-th processor operating frequency;
$v_{min} = v_1 \le v_2 \le ... \le v_M = v_{max}$;
$f_{min} = f_1 \le f_2 \le ... \le f_M = f_{max}$;
$1 \le m \le M$, M is the total number of processor operating points.

B. Energy Model

The energy consumption of modern processor for job execution, ξ, can be divided into two parts, dynamic energy consumption $\xi_{dynamic}$, and static energy consumption ξ_{static} [37]:

$$\xi = \xi_{dynamic} + \xi_{static} \tag{3}$$

According to [38], the dynamic power consumption $P_{dynamic}$ is computed as follows:

$$P_{dynamic} = A \cdot C \cdot v^2 \cdot f \tag{4}$$

Where,
A is the percentage of active gates;
C is the total capacitance load;
v is the supply voltage;
f is the processor frequency.

Then, we have:

$$\xi_{dynamic} = \sum_{\triangle t} P_{dynamic} \cdot \triangle t \tag{5}$$

where,
$P_{dynamic}$ is the dynamic power;
$\triangle t$ is a time period.

$\xi_{dynamic}$ is normally proportional to $E_{dynamic}$ [39], [40]:

$$\xi_{static} \propto \xi_{dynamic} \tag{6}$$

Therefore the whole power consumption could be estimated as follows [37]:

$$\xi \propto \xi_{dyanmic} \tag{7}$$

In conclusion, we have the performance model:

$$\xi = \sum_{\triangle t} (\delta \cdot v^2 \cdot f \cdot \triangle t) \tag{8}$$

Where,
δ is a constant determined by the PE.
v is the processor operating voltage during $\triangle t$;

f is the processor operating frequency during $\triangle t$;
$\triangle t$ is a time period.

C. Resource Model

A compute cluster normally contains multiple compute nodes, which are formally termed as Processing Elements (PEs) in a context of parallel computing. This paper makes a study on homogeneous clusters: all PEs of the cluster have the same processing speed or provide identical processing performance in term of MIPS (Million Instruction Per Second). A homogeneous cluster, C, contains K PEs. The k-th PE pe_k has two properties:

- $pe_k.v^{op} \in V$ is the processor operating voltage
- $pe_k.f^{op} \in F$ is the processor operating frequency

$1 \leq k \leq K$, K is the total number of PEs.

A cluster C is defined by its set of processing elements

$$C = \bigcup_{1 \leq k \leq K} \{pe_k\} \tag{9}$$

D. Parallel Task Model

A parallel task with precedence constrains is modeled as a Directed Acyclic Graph (DAG) – $T = (J, E)$:

- J: a set of jobs (nodes in a DAG)

  $$J = \bigcup_{1 \leq n \leq N} \{job_n\} \tag{10}$$

  where,
  job_n is a job in the parallel task J.
  N is the total number of jobs.
  A job, job_n, has 3 properties:
  - $weight$ is the instruction number of job_n.
  - t^{st} is the starting time of job_n.
  - t is the execution time of job_n. if job_n is executed on pe_k, the job execution time is calculated as follows:

    $$job_n.t = \frac{job_n.weight \cdot CPI}{pe_k.f^{op}} \tag{11}$$

    where, CPI is the number of cycles per instruction of pe_k. It is determined by both the hardware and software of the cluster C, for example, computer architecture and instruction set (ie, RISC or CISC). $job_n.t^0$ is the job_n's execution time when PE is running with the maximum frequency f_{max}. Equation 11 calculates job execution based on PE's operating frequency.
  - t^{end} is the ending time of job_n. We have:

    $$job_n.t^{end} = job_n.t^{st} + job_n.t \tag{12}$$

  Based on Equation 11 and Equation 8, the energy consumption to execute job_n can be calculated as follows:

  $$\xi_n = \gamma \cdot v^2 \cdot job_n.weight \tag{13}$$

  where, γ is a constant determined by the cluster C, and irrelevant with the parallel task T. v is the PE supply voltage during the job_n's execution.
- E: a set of precedence constraints (edges in a DAG)
  E defines partial orders (operational precedence constraints) on J. e_{ij} is an edge between job_i and job_j, it means that job_i must be completed before job_j can begin, $1 \leq i, j \leq N$, $job_i, job_j \in J$. e_{ij} sometime can also be represented $job_i < job_j$.
  e has one property:
  $e_{ij}.cost \geq 0$, is the amount of data required to be transferred from job_i to job_j, $1 \leq i, j \leq N$, $job_i, job_j \in J$. Data are transferred from the PE where job_i is executed to the PE where job_j is executed.
  As we are studying a homogeneous cluster, without loss of generality, $e_{i,j}.cost$ can also be normalized as communication time. Now we discuss the relationship between $e_{i,j}.cost$ and PE's operating frequency. Figure 2 shows the energy consumption and communication cost as processor frequency varies for four common MPI calls when different size of data are transferred among PEs [18]. From the experiment results we can see energy can be saved up to 31% with at most 5% communication time increase. In this paper, we ignore the communication time increase. In other words, when a PE's supplied voltage is scaled down, the data communication time remains unchanged.

V. Research Problem Definition

This paper focuses on two research issues:

- best effort scheduling
  The objective of scheduling is to minimize task execution time. Without damaging the performance of parallel task execution (task execution time), the best effort scheduling algorithm tries to reduce the energy consumption for task execution.
- energy-performance tradeoff scheduling
  The research issue is based on the green SLA negotiation between users and resource providers, which is discussed in Section III. Users agree to accept a tolerable performance loss, for example, additional 10% of task execution time, to reduce more energy consumption and make their computing more green. Therefore, the objective of the energy-performance tradeoff scheduling is to reduce energy consumption for task execution with an acceptable performance punishment.

Before we bring up the formal definition of the above research issues, the following term definitions are introduced.

- TST: Task Starting Time of T

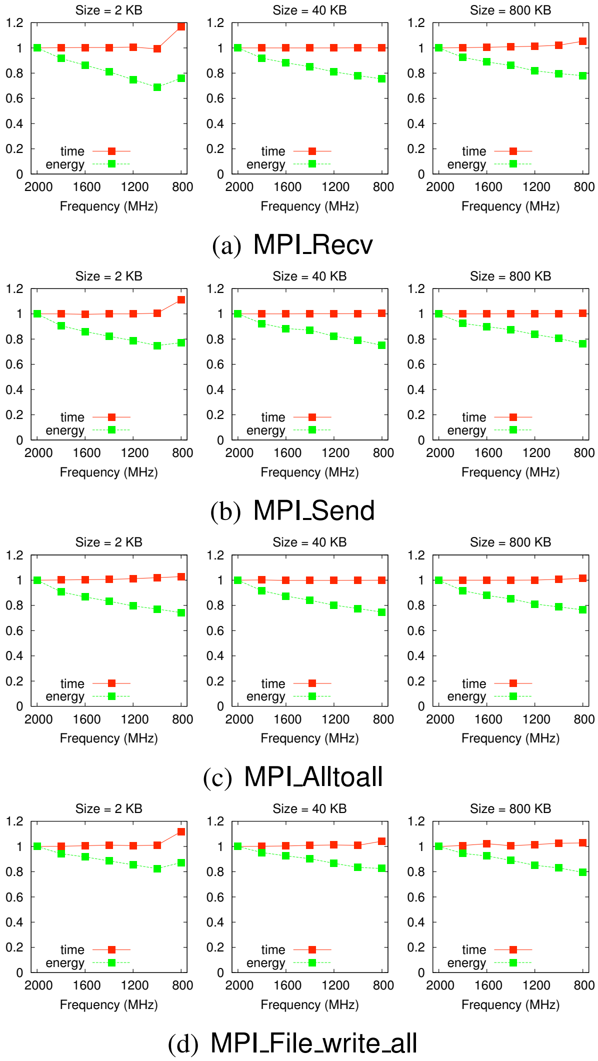
  $$TST = \min_{1 \leq n \leq N} job_n.t^{st} \tag{14}$$

- TFT: Task Finish Time of T

  $$TFT = \max_{1 \leq n \leq N} job_n.t^{end} \tag{15}$$

- $makespan$: the schedule length of T

  $$makespan = TFT - TST \tag{16}$$



(a) MPI_Recv

(b) MPI_Send

(c) MPI_Alltoall

(d) MPI_File_write_all

Fig. 2. Energy performance of MPI calls [18]

- $Schedule$: Task Schedule
  The $schedule_n$ of job_n is a mapping from job_n to a PE pe_k with task starting time $job_n.t^{st}$.

  $$schedule_n : job_n \rightarrow (pe_k, job_n.t^{st}) \tag{17}$$

  The schedule of parallel task T, $Schedule$, is defined as:

  $$Schedule = \bigcup_{1 \leq n \leq N} schedule_n \tag{18}$$

  A feasible schedule of parallel task T keeps the partial orders between jobs in T.

Based on the above definitions, the best effort scheduling issue is defined as: given parallel task T and a cluster C, find a feasible schedule $Schedule$, which 1) gives the minimum schedule length $makespan_{best}$ of T, and 2) reduce as much energy consumption as it can without increasing $makespan_{best}$.

The energy-performance tradeoff scheduling issue is defined as: given parallel task T, a cluster C, and the schedule length $makespan_{best}$, of a best effort schedule, find a feasible schedule which tries to minimize energy consumption by giving Task Execution Time $makespan \leq (1+\eta) \times makespan_{best}$. $\eta > 0$ is the accepted task execution time extension, which is determined by the green SLA negotiation.

VI. Power Aware Scheduling Algorithm for Parallel Tasks

A. Rules of Thumb for Scheduling

We summarize several obvious rules to guide the design of power aware scheduling algorithms for parallel tasks.

1) Equation 13 shows that given a certain task, a PE's supply voltage could be scaled down to a proper voltage to reduce the task's energy consumption. Certainly, this action may lead an increase of task execution time.
2) Figure 2 indicates that during the communication phase, the PE's supply voltage should be scaled down to the lowest level.
3) When a PE is idle (there is no task execution and data communication), its supply voltage should be leveled down to the lowest level.

B. Best Effort Scheduling Algorithm

This subsection discusses the best effort scheduling algorithm, which aims to minimize energy consumption of task execution without extending a task's makespan. The best effort scheduling algorithm (shown in Algorithm 1) firstly employs the ETF (Earliest Task First), a list-based scheduling algorithm (shown in Algorithm 2), to find the best effort task response time for T. Then, it tries to reduce the energy consumption with the following methods:

- scale down PE's voltages to a proper level, thus extending the execution time of the non-critical jobs without affecting the critical path.
- scale the PE's voltage when it is idle or when it is in the data communication phase.

Algorithm 1 Best effort power aware scheduling algorithm

```
1. schedule tasks via the ETF scheduling algorithm 2
2. scale down PE's voltages for all non-critical jobs with Algorithm 3
```

Given a parallel task T, the ETF algorithm [41], [42] is described in Algorithm 2. The Algorithm 2 allocates each job with a priority which be calculated via different methods, for example, bottom level and top level [43], [44]. In our implementation, we use the bottom level. Then, Algorithm 2 selects ready jobs with the highest priority and schedules it on the PE with earliest task starting time.

The output of Algorithm 2 is a so-called Gantt chart, which displays time slots for job execution and data communication on multiple PEs. Figure 3 is an example parallel task to be scheduled. In Figure 3, job IDs and job execution costs are marked inside the jobs and the communication costs are labeled on the links. The scheduled task graph is shown in

Algorithm 2 The ETF scheduling algorithm

```
$job_n.level$: priority of task $job_n \in J$
$ready\_job\_list$: list of jobs that are ready to be executed
$PE\_list$: list of PEs
$pe_k.t^{available}$: PE's available time.

BEGIN

FOR each job $job_n \in J$ DO
        compute $job_n.level$
ENDFOR

put all ready jobs into $ready\_job\_list$
sort all jobs $job_n \in ready\_job\_list$ in decreasing order of $job_n.level$

put all PEs into $PE\_list$
sort all PEs $pe_k.t^{available} = 0$

REPEAT
  IF ($ready\_job\_list \neq \emptyset$ ) THEN
    get a job, $job_n$, from $ready\_job\_list$
    get a PE, $pe_k$, which has the earliest available time $pe_k.t^{available}$
    schedule $job_n$ on $pe_k$
    arrange the communicate phase, calculate starting time and finish time of $job_n$ on $pe_k$
    delete the task from $ready\_job\_list$
    update $PE\_list$ with increasing order $pe_k.t^{available}$
  ENDIF
  update $ready\_job\_list$
UNTIL (every job $job_n \in J$ has been scheduled)

END
```

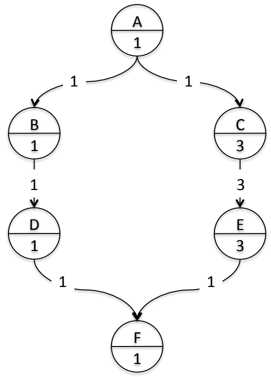


Fig. 3. An example DAG

Figure 4 as a Gantt chart. The Critical Path (CP) of scheduled task graph in a Gantt chart is defined as follows:

> a set of time slots of job execution and data communication from the first job to the last job, of which the sum of computation costs and communication costs is the $makespan$.

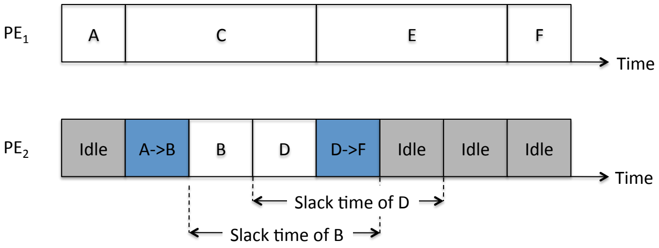


Fig. 4. An example Gantt chart

The CP in Figure 4 is "$A \rightarrow C \rightarrow E \rightarrow F$". It should be aware that a CP may across multiple PEs. As the best effort scheduling algorithm does not extend the $makespan$, supplied voltages of PEs during the time slots of task execution and data communication in the CP is not changed. Supplied voltages of other time slots in a Gantt chart are considered be scaled down. For example, in Figure 4 jobs B and D have chance to extend their execution time and scale down supplied voltages.

To discuss the Algorithm for scaling voltages on non-critical time slots, we need to compute the slack time for a non-critical job. We have job_n's earliest start time is:

$$job_n.\overleftarrow{t^{st}} = \max_{\{job_m|job_m<job_n\}} \{job_m.t^{end} + e_{m,n}.cost\} \tag{19}$$

job_n's latest finish time is:

$$job_n.\overrightarrow{t^{end}} = \min_{\{job_l|job_l>job_n\}} \{job_l.t^{st} - e_{l,n}.cost\} \tag{20}$$

$\{job_m|job_m < job_n\}$ and $\{job_l|job_l > job_n\}$ are job_n's precursor set and successor set respectively. Then job_n's slack time can be calculated as:

$$job_n.slack = job_n.\overrightarrow{t^{end}} - job_n.\overleftarrow{t^{st}} \tag{21}$$

We can find in Figure 4 the slack time of job B and D.

Assume job_n is a non-critical job and is executed on pe_k. Then job_n's execution time can be extended to $job_n.slack$ without violating precedence constraints (without changing the finish time of its precursors and the start time of its successors). pe_k's operating frequency can be scaled to $pe_k.f^{op}$,

$$pe_k.f^{op} = f_{max} \times \frac{job_n.t^0}{job_n.slack} \tag{22}$$

Where, $job_n.t^0$ is job_n's execution time when pe_k is operated with f_{mx}. $job_n.t^0$ is discussed in Section IV-D and can be calculated in Equation 11.

Algorithm 3 shows how to scale down non-critical jobs. For each PE, it scans all time slots (line 2–3). When the PE is idle or transfers data in a time slot, Algorithm 3 scales the PE's operating frequency to the lowest (line 4–6). When a time slot executes a non-critical job, it calculates its slack time, extends the job's execution time to the slack time, and scales down the PE's operating frequency to a proper value (line 7–9).

C. Energy-Performance Tradeoff Scheduling Algorithm

Now we discuss the energy-performance tradeoff problem: if a user agrees to tolerate an increase of his/her job execution time, for example, η of schedule length of the best-effort

Algorithm 3 Non-critical time slot voltage scaling algorithm

```
BEGIN

FOR each PE $pe_k$ DO
  FOR each time slot in $pe_k$'s Gantt chart DO
    IF $pe_k$ is idle or it executes a communication phase THEN
      scale down $pe_k$'s operating frequency to lowest
    ENDIF

    IF $pe_k$ executes a non-critical job $job_n$ THEN
      calculate $job_n.slack$ as Equation 21.
    scale $pe_k$'s frequency to $pe_k.f^{op}$ as Equation 22.
     ENDIF
   ENDFOR
ENDFOR

END
```

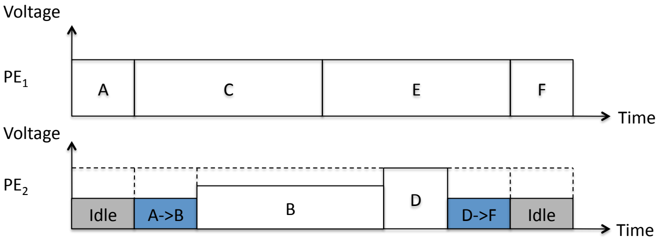


Fig. 5. Example power Gantt chart

scheduling algorithm, how to schedule jobs to save more energy?

The energy-performance tradeoff algorithm is shown in Algorithm 4. It firstly gets the best effort scheduling length via Algorithm 2. Then, it scales both the critical time slots in Algorithm 5 and non-critical time slots in Algorithm 3.

Algorithm 4 Energy-performance tradeoff scheduling algorithm

1. schedule tasks via the ETF scheduling algorithm 2
2. scale down PE's voltages for critical jobs with Algorithm 5
3. scale down PE's voltages for non-critical jobs with Algorithm 3

The Algorithm 5 firstly extends the critical time slots. Assume job_n is a critical job and it is executed on pe_k. It has been proved in [45] that distributing the free slack time evenly is optimal as the power consumption is a convex function of PE frequency. Therefore job_n's slack time can be calculated as:

$$job_n.slack = job_n.t^0 \times \eta \qquad (23)$$

Where,
$job_n.t^0$ is job_n's execution time when pe_k is operated with f_{max}.
η is the agreed extension of parallel task's execution time.

pe_k's operating frequency can be scaled to $pe_k.f^{op}$,

$$pe_k.f^{op} = f_{max} \times \frac{job_n.t^0}{job_n.slack} \qquad (24)$$

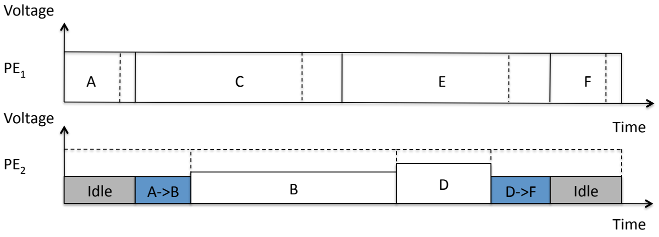


Fig. 6. Energy-performance tradeoff power Gantt chart

Algorithm 5 Algorithm of voltage scaling for all time slots

```
BEGIN

FOR each PE $pe_k$ DO
   FOR each time slot in $pe_k$'s Gantt chart DO
     IF $pe_k$ executes a critical job $job_n$ THEN
   calculate its $job_n$'s slack time as Equation 23
   scale $pe_k$'s frequency to $pe_k.f^{op}$ as Equation 24.
   ENDFOR

   FOR each time slot in $pe_k$'s Gantt chart DO
     IF $pe_k$ is idle or it executes a communication phase THEN
       scale down $pe_k$'s operating frequency to lowest
     ENDIF

     IF $pe_k$ executes a non-critical job $job_n$ THEN
       calculate $job_n.slack$ as Equation 21.
     scale $pe_k$'s frequency to $pe_k.f^{op}$ as Equation 22.
      ENDIF
    ENDFOR
ENDFOR

END
```

VII. Performance Study With Simulation

We make a simulation study on the proposed best effort scheduling algorithm and energy-performance tradeoff scheduling algorithm. Several task sets are generated with the Synthetic DAG generation tool [46]. We simulate a cluster with multiple Turion MT-34 processors, whose operating points are shown in Table I.

In this simulation for best effort scheduling, we are interested how much energy is saved given various parallel tasks and PE numbers in the cluster. We define the resource competition to execute a parallel task, $\zeta(T)$, in a cluster as follows:

$$\zeta(T) = \frac{N}{P} \qquad (25)$$

where, T is the parallel task, N is the job number of T, and P is the PE number for executing T. Resource competition shows

TABLE I
OPERATING POINTS FOR THE TURION MT-34 PROCESSOR

Frequency (GHz)	Supply Voltage (V)
1.8	1.20
1.6	1.15
1.4	1.10
1.2	1.05
1.0	1.00
0.8	0.90

the task execution situation, like how many precedences exist between jobs, how many jobs are scheduled, and how many jobs are executed on each PE.

TABLE II
COMPARISON OF ENERGY SAVINGS BETWEEN DIFFERENT ENERGY AWARE SCHEDULING ALGORITHM

Energy aware DAG scheduling algorithm	Maximum energy saving
EADUS & TEBUS [28]	16.8%
Energy Reduction Algorithm [31]	25%
LEneS [22]	28%
ECS [30]	38%
Our algorithm	44.3%

Our best effort scheduling algorithm can achieve up to 44.3% energy saving in the simulation. Table II compares our algorithm with other energy aware DAG scheduling algorithms in term of max energy saving. EADUS & TEBUS [28] uses the duplication strategies for scheduling DAG based parallel tasks in a cluster to reduce power consumption. However, EADUS & TEBUS do not use DVFS to reduce energy consumption, thus leading less energy savings. Compared with LEneS [22], Energy Reduction Algorithm [31], and ECS [30], our algorithm can achieve more energy saving as 1) it reduces the energy consumption during the communication phase, 2) it reduces power consumption when a PE is idle, and 3) it tries to extend job slack time whenever it is possible.

Figure 7 shows the energy savings in different scenarios of numbers of PEs and resource competition. From Figure 7 we can see that the energy saved increases as the number of PEs increases. This can be explained as follows: when the number of PEs increases, intuitively there are less jobs executed in a PE, then the jobs have more of a chance to scale their execution time and PE supply voltages. If we fix the number of PEs, the energy saving firstly increases, achieves it maximum value, and then deceases. This can be explained by the fact that the percentage of jobs on the critical path firstly increases then decrease. The length of critical path gives the limit that non-critical jobs can extend to.

In the simulation for enery-performance tradeoff scheduling, we are more interested in the relationship between the energy saved and the extended task execution time, as shown in Figure 8. From Figure 8 we can see that:

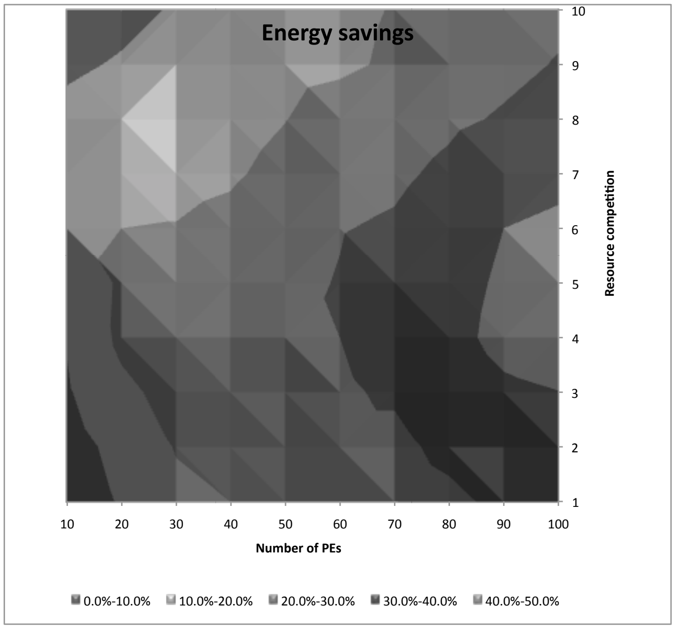


Fig. 7. Energy savings of best effort scheduling algorithm

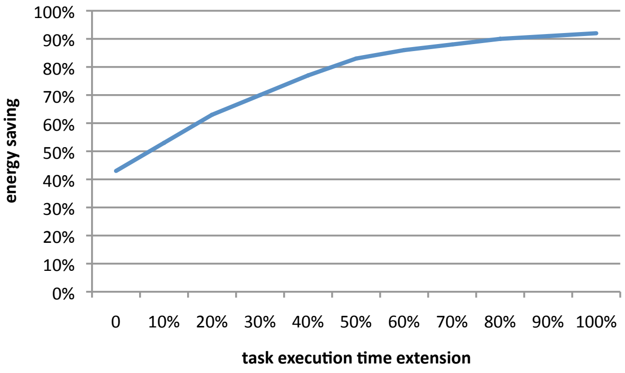


Fig. 8. Energy savings vs. makespan extension

- When the makespan extension increases, the energy savings also increase.
- Then energy savings increase much when the makespan extension is less then 30%.
- The energy savings become saturated when the maskspan extension is more than 70%.

These observations can conclude that the green SLA negotiation is feasible. When users pay additional tolerant task execution time, which is less than 30%, significant energy savings can be achieved. This is a win-win game.

VIII. CONCLUSION AND FUTURE WORK

Recently, the need for efficient algorithms to minimize wasted server energy has become increasingly important. Dynamic voltage and frequency scaling (DVFS) technique has proven to be a highly effective technique to achieve low power consumption for high performance computing by dynamically scaling processor speed. We develop our research on minimizing energy for precedence-constrained parallel task execution. This paper proposes a scheduling algorithm in DFVS-enabled clusters for executing parallel tasks. The proposed algorithm finds slack time for non-critical jobs without increasing scheduling length. We also develop green SLA based mechanism to reduce energy consumption by return users tolerant increased scheduling makespans. The proposed scheduling algorithm is examined via a simulation study. Test results show that the scheduling algorithm is efficient to reduce the power consumption of a DVFS-enabled cluster. Future work includes the deployment of the power aware scheduling algorithm in some real applications, for example, the the sparse Cholesky decomposition.

References

[1] G. von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-Aware Scheduling of Virtual Machines in DVFS-enabled Clusters," in *IEEE Cluster 2009*. New Orleans: IEEE, 31 Aug. – Sep. 4 2009. [Online]. Available: http://code.google.com/p/cyberaide/source/browse/trunk/papers/09-greenit-cluster09/vonLaszewski-cluster09.pdf

[2] L. Wang, G. von Laszewski, J. Dayal, X. He, and T. R. Furlani, "Thermal Aware Workload Scheduling with Backfilling for Green Data Centers," in *Proceedings of the 28th IEEE International Performance Computing and Communications Conference*, Arizona, U.S., Dec 2009.

[3] W. Forrest, "How to cut data centre carbon emissions?" Website, December 2008. [Online]. Available: http://www.computerweekly.com/Articles/2008/12/05/233748/how-to-cut-data-centre-carbon-emissions.htm

[4] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Trans. Computers*, vol. 37, no. 11, pp. 1384–1397, 1988.

[5] V. Sarkar, *Partitioning and scheduling parallel programs for multiprocessors*. Cambridge, MA, USA: MIT Press, 1989.

[6] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Trans. Software Eng.*, vol. 12, no. 5, pp. 662–675, 1986.

[7] Y. Wang and R. J. T. Morris, "Load sharing in distributed systems," *IEEE Trans. Computers*, vol. 34, no. 3, pp. 204–217, 1985.

[8] R. Li and H. Huang, "List scheduling for jobs with arbitrary release times and similar lengths," *J. Scheduling*, vol. 10, no. 6, pp. 365–373, 2007.

[9] A. Mtibaa, B. Ouni, and M. Abid, "An efficient list scheduling algorithm for time placement problem," *Computers & Electrical Engineering*, vol. 33, no. 4, pp. 285–298, 2007.

[10] C.-H. Hsu and W. chun Feng, "A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters," in *CLUSTER*, 2005, pp. 1–10.

[11] C. Hsu and W. Feng, "A power-aware run-time system for high-performance computing," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Washington, DC, USA, 2005.

[12] I. Gorton, Greenfield, P., Szalay, A., and R. Williams, "Data-Intensive Computing in the 21st Century," *IEEE Computer*, vol. 41, no. 4, pp. 30–32, 2008.

[13] W. chun Feng, A. Ching, and C.-H. Hsu, "Green supercomputing in a desktop box," in *Proceedings of the 21th International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007, pp. 1–8.

[14] W. chun Feng and T. Scogland, "The green500 list: Year one," in *Proceedings of the 23rd IEEE International Symposium on Parallel and Distributed Processing*, 2009, pp. 1–7.

[15] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan, "Reducing power with performance constraints for parallel sparse applications," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*. Washington, DC, USA: IEEE Computer Society, 2005, p. 231.1.

[16] R. Ge, F. X., F. W., and K. Cameron, "CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters," in *Proceedings of the 2007 International Conference on Parallel Processing*. IEEE Computer Society Washington, DC, USA, 2007.

[17] V. Freeh and D. Lowenthal, "Using multiple energy gears in MPI programs on a power-scalable cluster," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM New York, NY, USA, 2005, pp. 164–173.

[18] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs," in *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2006, p. 107.

[19] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1995, p. 374.

[20] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," in *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2001, pp. 279–282.

[21] G. yeon Wei, J. Kim, D. Liu, S. Sidiropoulos, and M. A. Horowitz, "A variable-frequency parallel i/o interface with adaptive power-supply regulation," *IEEE J. Solid-State Circuits*, vol. 35, pp. 1600–1610, 2000.

[22] F. Gruian and K. Kuchcinski, "Lenes: task scheduling for low-energy systems using variable supply voltage processors," in *Proceedings of Asia and South Pacific Design Automation Conference*, 2001, pp. 449–455.

[23] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, Nov. 2002, pp. 721–725.

[24] J. Luo and N. Jha, "Power-efficient scheduling for heterogeneous distributed real-time embedded systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 6, pp. 1161–1170, June 2007.

[25] J. Luo, N. K. Jha, and L.-S. Peh, "Simultaneous dynamic voltage scaling of processors and communication links in real-time distributed embedded systems," *IEEE Trans. VLSI Syst.*, vol. 15, no. 4, pp. 427–437, 2007.

[26] Y. Zhang, X. S. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th annual Design Automation Conference*. New York, NY, USA: ACM, 2002, pp. 183–188.

[27] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of dvs processing elements for energy minimisation in distributed systems," in *In Proc. ISSS*, 2001, pp. 250–255.

[28] Z. Zong, A. Manzanares, B. Stinar, and X. Qin, "Energy-aware duplication strategies for scheduling precedence-constrained parallel tasks on clusters," in *Proceedings of the 2006 IEEE International Conference on Cluster Computing*, 2006.

[29] S. Baskiyar and K. K. Palli, "Low power scheduling of dags to minimize finish times," in *13th International Conference on High Performance Computing*, 2006, pp. 353–362.

[30] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 92–99.

[31] H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi, "Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster," *Cluster Computing, IEEE International Conference on*, vol. 0, pp. 1–10, 2006.

[32] G. von Laszewski and L. Wang, "GreenIT Service Level Agreements," in *Service Level Agreements in Grids Workshop, colocated with IEEE/ACM Grid 2009 Conference*, Banff, Canada, 13 Oct. 2009. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/09-greenit-sla/vonLaszewski-greenit-sla.pdf

[33] G. Verdun, "The Green Grid metrics: Data center infrastructure efficiency (DCiE) detailed anaysis," The Green Grid, Tech. Rep., Feb. 2007.

[34] C. Belady, "The Green Grid Data center Efficiency Metrics: PUE and DCIE," The Green Grid, Tech. Rep., Feb. 2007.

[35] J. H. et. al., "A Framework for Data Center Energy Productivity," The Green Grid, Tech. Rep., Feb. 2008.

[36] "SWaP (Space, Watts and Performance) Metric," Web Page. [Online]. Available: http://www.sun.com/servers/coolthreads/swap/

[37] K. H. Kim, R. Buyya, and J. Kim, "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters," in *CCGRID*, 2007, pp. 541–548.

[38] R. Ge, X. Feng, and K. Cameron, "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Washington, DC, USA, 2005.

[39] J. Li and J. F. Martínez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *HPCA*, 2006, pp. 77–

87.

[40] C. Piguet, C. Schuster, and J. Nagel, "Optimizing architecture activity and logic depth for static and dynamic power reduction," in *Circuits and Systems, 2004. NEWCAS 2004. The 2nd Annual IEEE Northeast Workshop on*, 2004, pp. 41–44.

[41] B. A. Shirazi, K. M. Kavi, and A. R. Hurson, Eds., *Scheduling and Load Balancing in Parallel and Distributed Systems*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1995.

[42] Q. Wang and K. H. Cheng, "List scheduling of parallel tasks," *Inf. Process. Lett.*, vol. 37, no. 5, pp. 291–297, 1991.

[43] I. Ahmad, Y.-K. Kwok, and M.-Y. Wu, "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors," in *Proceedings of the 1996 International Symposium on Parallel Architectures, Algorithms and Networks*. Washington, DC, USA: IEEE Computer Society, 1996, p. 207.

[44] T. L. Adam, K. M. Chandy, and J. R. Dickson, "A comparison of list schedules for parallel processing systems," *Commun. ACM*, vol. 17, no. 12, pp. 685–690, 1974.

[45] J. Luo and N. K. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," in *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*. Washington, DC, USA: IEEE Computer Society, 2002, p. 719.

[46] F. SUTER, "Synthetic dag generation," Web Page. [Online]. Available: http://www.loria.fr/∼suter/dags.html